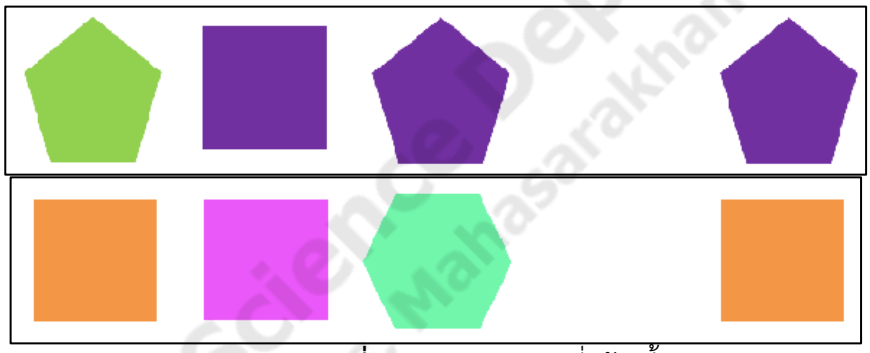
บทที่ 3
กระบวนการดำเนินงาน

3.1 ศึกษาข้อมูลที่เกี่ยวข้อง

เริ่มจากการศึกษาทำความเข้าใจในส่วนของโปรแกรม Matlab เพื่อใช้พัฒนาโปรแกรม นอกจากนี้ยังต้องทำความเข้าใจเกี่ยวกับเนื้อหาเรื่อง Deep Learning เพื่อนำมาเป็นแนวคิดในการทำโปรแกรม

3.2 การจัดทำชุดข้อมูลที่ใช้ในการทดลอง

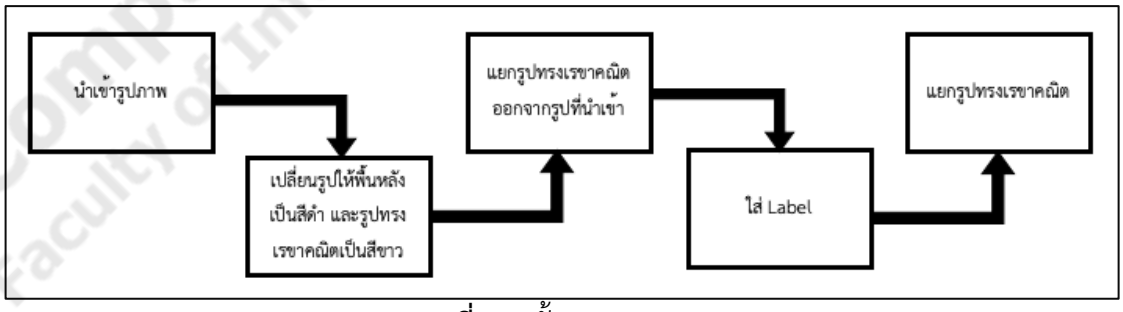
ทำการหาภาพเพื่อนำมาใช้เป็นชุดข้อมูลในการทดสอบและพัฒนาระบบ โดยรูปที่คัดเลือกมาจะเป็นรูปที่สร้างขึ้นมาเอง ดังภาพประกอบที่ 3.1



ภาพประกอบที่ 3.1 ตัวอย่างภาพที่สร้างขึ้น

3.3 การตรวจสอบภาพ

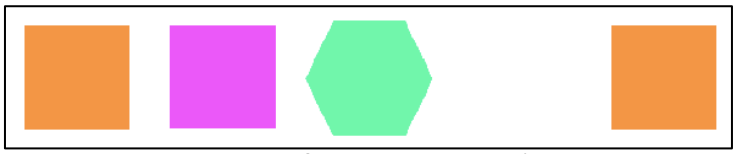
เป็นการแบ่งแยกรูปเรขาคณิต โดยมีขั้นตอน ดังภาพประกอบที่ 3.2



ภาพประกอบที่ 3.2 ขั้นตอนการตรวจสอบภาพ

3.3.1 การนำเข้าของรูปภาพ

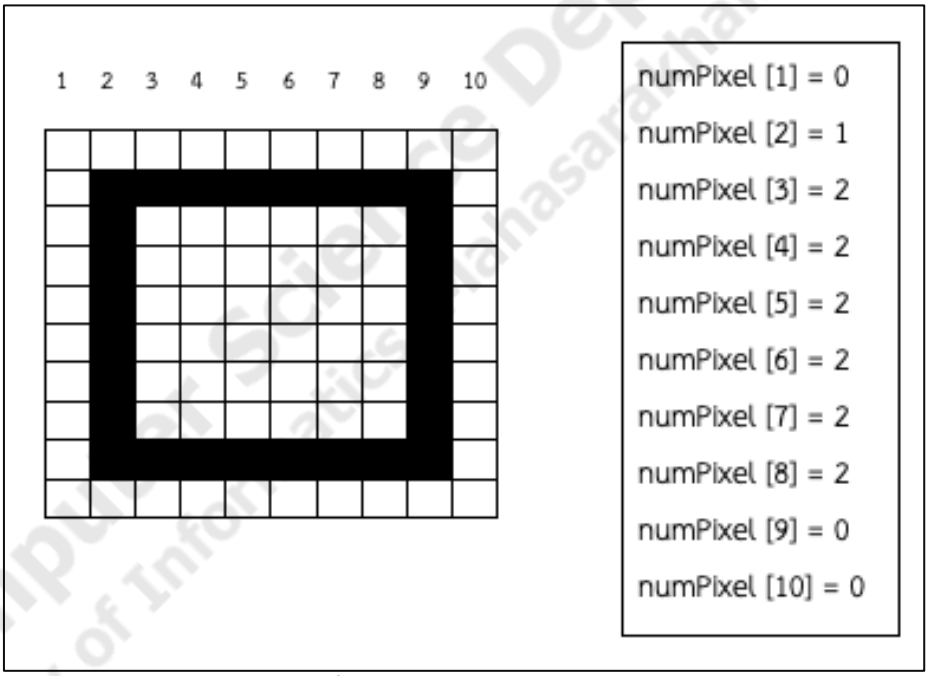
เป็นการเลือกภาพที่ต้องการแยกหารูปเรขาคณิต ดังภาพประกอบที่ 3.3



ภาพประกอบที่ 3.3 ตัวอย่างภาพที่นำเข้า

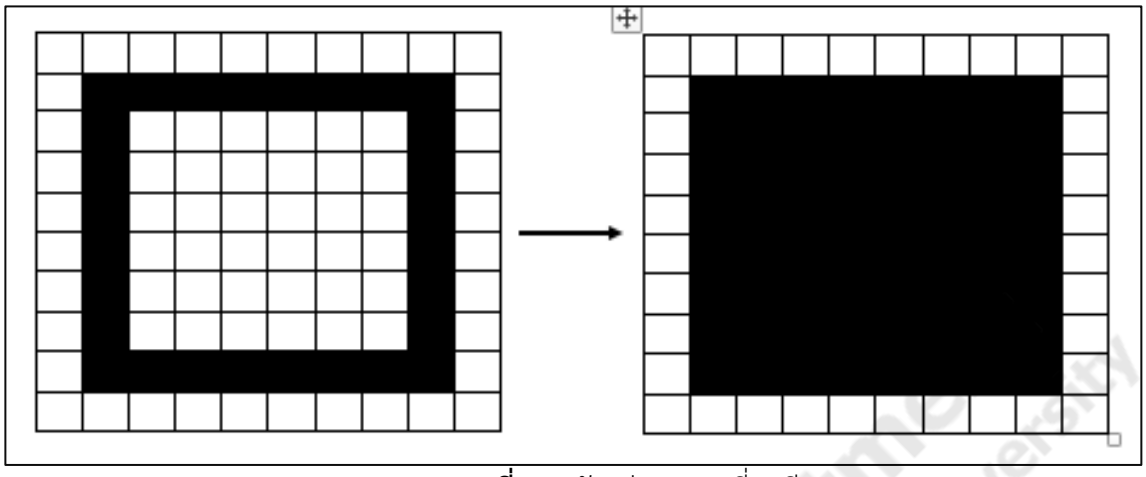
3.3.2 การเปลี่ยนพื้นหลังเป็นสีดำ และรูปทรงเรขาคณิตเป็นสีขาว

การเปลี่ยนสีพื้นหลังและรูปทรงเรขาคณิต เกิดจากการนับจำนวนของสีดำแต่ละพิกเซลของแนวตั้ง โดยจะกำหนดค่า numPixel เป็นอาเรย์เท่ากับจำนวนของคอลัมน์ภาพที่นำเข้า และเก็บค่าทีละอาเรย์ตามจำนวนสีดำที่ตรวจเจอ เพื่อนำค่ามาตรวจสอบว่ามีพื้นที่ว่างระหว่างแถวที่คอลัมน์นั้นหรือไม่ ดังภาพประกอบที่ 3.4



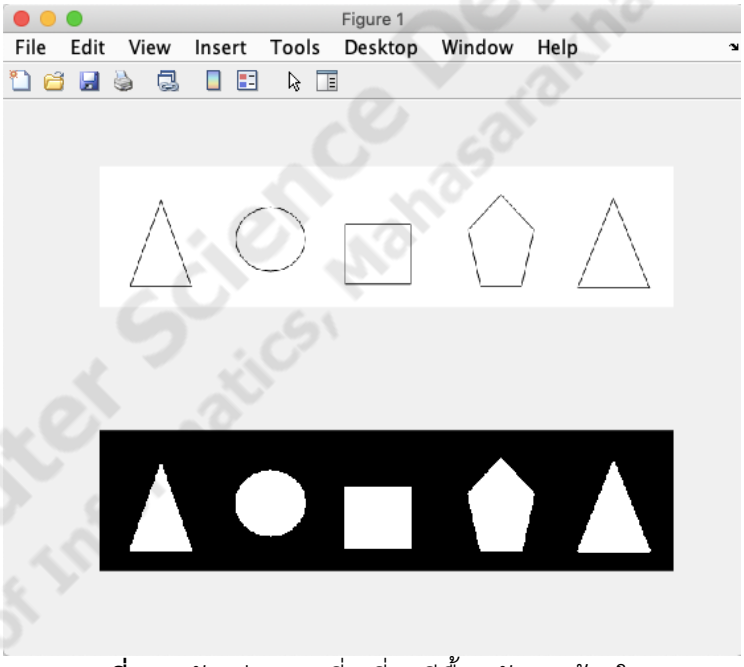
ภาพประกอบที่ 3.4 ตัวอย่างการนับสีดำแต่ละพิกเซล

จากภาพประกอบที่ 3.4 จะได้ค่า numPixel มาเพื่อเปลี่ยนสีพื้นหลังและรูปทรงเรขาคณิต ถ้าค่า numPixel มีค่าเท่ากับ 2 ให้เปลี่ยนสีเมื่อ ตรวจเจอสีดำและเปลี่ยนสีพิกเซลถัดไปที่เป็นสีขาวให้เป็นสีดำไปเรื่อย ๆ จากกว่าจะตรวจเจอพิกเซลสีเป็นสีดำ ดังภาพประกอบที่ 3.5



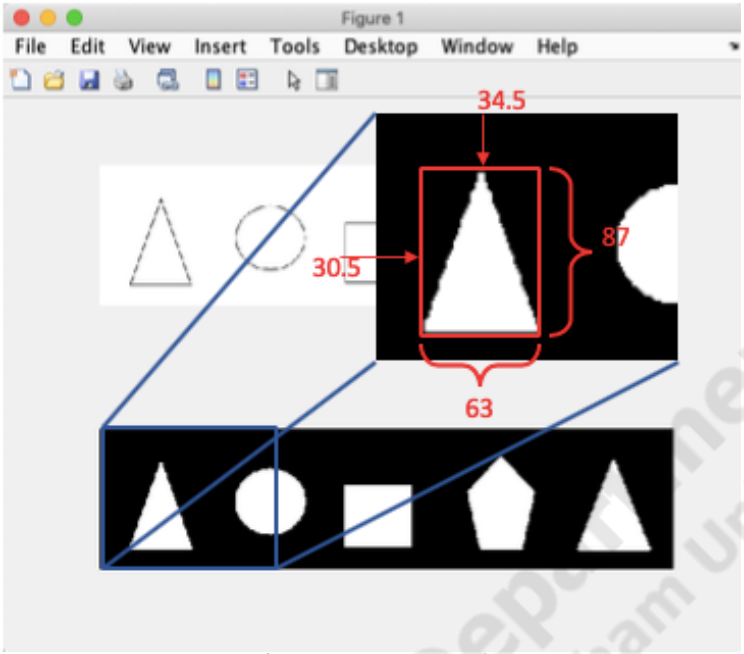
ภาพประกอบที่ 3.5 ตัวอย่างการเปลี่ยนสี

เมื่อได้ภาพที่ออกมา ดังภาพประกอบที่ 3.5 ขั้นตอนต่อไปเป็นการเปลี่ยนสีทั้งหมดจากสีดำเป็นสีขาว และจากสีขาวเป็นสีดำ เพื่อให้ง่ายต่อการตรวจจับรูปทรงเรขาคณิต ดังภาพประกอบที่ 3.6



ภาพประกอบที่ 3.6 ตัวอย่างภาพที่เปลี่ยนสีพื้นหลังและด้านในรูปเรขาคณิต

จากภาพประกอบที่ 3.6 สามารถหาตำแหน่งของรูปต่าง ๆ ได้ในภาพ โดยกำหนดให้ rect เป็นตำแหน่งที่โปรแกรมสามารถตัดภาพออกมาได้ เช่น rect = [30.5,34.5,63,87] (30.5คือความห่างระหว่างขอบของภาพด้านซ้ายไปจนถึงจุดแรกด้านซ้ายของรูปทรงเรขาคณิต ,34.5คือความห่างระหว่างขอบของภาพด้ายบนไปจนถึงจุดแรกด้านบนของรูปทรงเรขาคณิต ,63คือความกว้างของรูปทรงเรขาคณิต และ87คือความสูงของรูปทรงเรขาคณิต) ดังภาพประกอบที่ 3.7



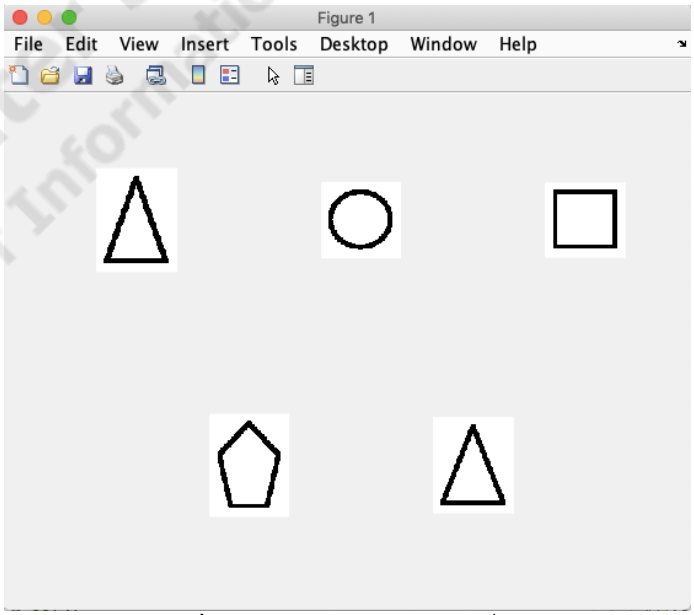
ภาพประกอบที่ 3.7 ตัวอย่างภาพที่หาตำแหน่งได้

3.3.3 การแยกรูปทรงเรขาคณิตออกจากรูปที่นำเข้า

จากขั้นตอนข้างต้น สามารถหาตำแหน่งของรูปทรงเรขาคณิตได้ครบทุกรูปแบบ และสามารถแยกออกจากภาพได้

3.3.4 ใส่ Label

ดังภาพประกอบที่ 3.8



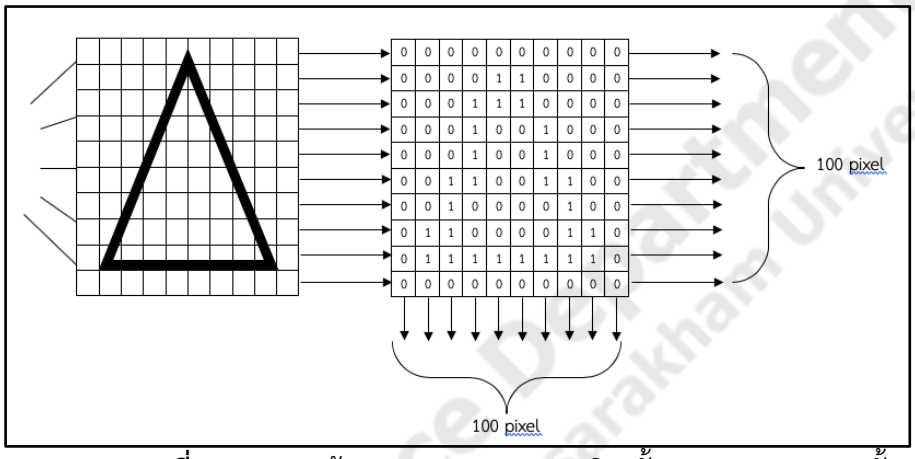
ภาพประกอบที่ 3.8 ตัวอย่างการนำภาพที่แยก ใส่ Label

3.3.5 การแยกรูปทรงเรขาคณิต

ขั้นตอนนี้ เป็นขั้นตอนที่ตอนทำงานกับกระบวนการจำแนก

3.4 กระบวนการจำแนก

หลังจากได้รูปทรงเรขาคณิตจากการแยกตัวอักษรแต่ละตัวก็ทำการย่อภาพให้เหลือขนาด 64x64 พิกเซล ดังภาพประกอบที่ 3.9 เพื่อนำไประบุว่าเป็นรูปทรงเรขาคณิต โดยใช้โครงข่ายประสาทเทียมแบบสังวัตนาการ (Convolutional Neural Network ; CNN) จากภาพต้นฉบับ 60,000 ภาพ แบ่งเป็นข้อมูลในการเรียนรู้ 60% คิดเป็น 36,000 ภาพ และข้อมูลในการทดสอบ 40% คิดเป็น 24,000 ภาพ



ภาพประกอบที่ 3.9 การหาข้อมูลของรูปทรงเรขาคณิต ทั้งแนวนอน และแนวตั้ง

จากภาพประกอบที่ 3.9 สามารถหา input of size ได้จากการนำจำนวนพิกเซลแถวและหลักมาคูณกัน จะได้ 64*64 = 4096 (input of size มีค่าเท่ากับขนาดของภาพ)และเพื่อหา output of size

$$\frac{N - F + 2P}{S} + 1$$

N = ขนาดของภาพ

F = ขนาดของ Filter

P = จำนวนของ Padding

S = จำนวนของ Stride

3	3	3	3	3	3
3	2	2	2	2	3
3	2	1	1	2	3
3	2	1	1	2	3
3	2	2	2	2	3
3	3	3	3	3	3

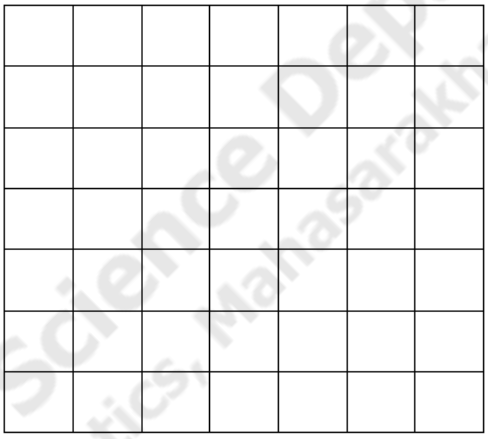
ภาพประกอบที่ 3.10 ตัวอย่างภาพที่input มาขนาด 6*6

1	0
0	1

ภาพประกอบที่ 3.11 ตัวอย่าง Filter ที่กำหนดขนาด 2*2

การคำนวณค่าใหม่ทำโดยการกำหนดค่า Filter ขนาด 2*2 ดังภาพประกอบที่ 3.11 จากนั้นกำหนด Stride = 1 (Stride คือ จำนวนของการขยับ Filter) และกำหนด Padding = 1 จากนั้นแทนค่าในสมการ output of size เพื่อหาขนาดของภาพใหม่หลังจากการทำ Convolution แล้วแทนค่าในสมการข้างต้น

$$\text{output of size} = \frac{6 - 2 + 2(1)}{1} + 1 = 7$$



ภาพประกอบที่ 3.12 ตัวอย่างขนาดของภาพใหม่ ขนาด 7*7

เนื่องจาก Padding = 1 ที่ให้เราต้องเพิ่มค่าขอบของภาพ 6*6 เป็น 0 ทุกช่องเพื่อที่จะให้ Filter วาง และ Stride ไปได้ ดังภาพประกอบที่ 3.12

0	0	0	0	0	0	0	0
0	3	3	3	3	3	3	0
0	3	2	2	2	2	3	0
0	3	2	1	1	2	3	0
0	3	2	1	1	2	3	0
0	3	2	2	2	2	3	0
0	3	3	3	3	3	3	0
0	0	0	0	0	0	0	0

ภาพประกอบที่ 3.13 ตัวอย่างภาพ input ที่ทำการเพิ่มขอบของภาพต้นฉบับ

0	0	0	0	0	0	0	0
0	3	3	3	3	3	3	0
0	3	2	2	2	2	3	0
0	3	2	1	1	2	3	0
0	3	2	1	1	2	3	0
0	3	2	2	2	2	3	0
0	3	3	3	3	3	3	0
0	0	0	0	0	0	0	0

ภาพประกอบที่ 3.14 ตำแหน่งการวาง Filter ในรอบแรก

จานนั้นจะทำการ วาง Filter บนภาพประกอบที่ 3.14 จะทำการคูณค่าในตำแหน่งที่ตรงกันของ Filter และนำค่าที่คูณมาบวกกัน ผลลัพธ์ที่ได้จากการบวก หมายถึงค่าใหม่ที่จะใช้แทนค่าในขนาดของภาพใหม่ที่คำนวณไว้ก่อนหน้า

รอบที่1 การคำนวณหาค่าใหม่ทำได้โดยเอาค่าในภาพคูณกับค่าใน Filter ที่ตำแหน่งตรงกันทำจนครบขนาดของ Filter การคำนวณผลลัพธ์ของการ Stride ในรอบที่1

ตำแหน่งที่1 = 0*1 = 0

ตำแหน่งที่2 = 0*0 = 0

ตำแหน่งที่3 = 0*0 = 0

ตำแหน่งที่4 = 3*1 = 3

จากนั้น หาผลรวมของทุกตำแหน่ง (ตำแหน่งที่1 +...+ ตำแหน่งที่4) = 3 และแทนค่าในตำแหน่งที่ 1 ดังภาพประกอบที่ 3.15 ขยับไปทีละ 1 ช่อง

3						

ภาพประกอบที่ 3.15 ค่าใหม่หลังจากการ Stride ในรอบที่ 1

0	0	0	0	0	0	0	0
0	3	3	3	3	3	3	0
0	3	2	2	2	2	3	0
0	3	2	1	1	2	3	0
0	3	2	1	1	2	3	0
0	3	2	2	2	2	3	0
0	3	3	3	3	3	3	0
0	0	0	0	0	0	0	0

ภาพประกอบที่ 3.16 ตำแหน่งการวาง Filter ในรอบที่ 2

รอบที่2 การคำนวณหาค่าใหม่ทำได้โดยเอาค่าในภาพคูณกับค่าใน Filter ที่ตำแหน่งตรงกันทำจนครบขนาดของ Filter การคำนวณผลลัพธ์ของการ Stride ในรอบที่2

ตำแหน่งที่1 = 0*1 = 0

ตำแหน่งที่2 = 0*0 = 0

ตำแหน่งที่3 = 2*0 = 0

ตำแหน่งที่4 = 3*1 = 3

จากนั้น หาผลรวมของทุกตำแหน่ง (ตำแหน่งที่1 +...+ ตำแหน่งที่4) = 3 ตำแหน่งที่ 2 ดังภาพประกอบที่ 3.14 ขยับไปทีละ 1 ช่องและหาค่าจนครบทุกช่องที่ขนาด 7*7 ผลลัพธ์ที่ได้จะปรากฏบนภาพประกอบที่ 3.18

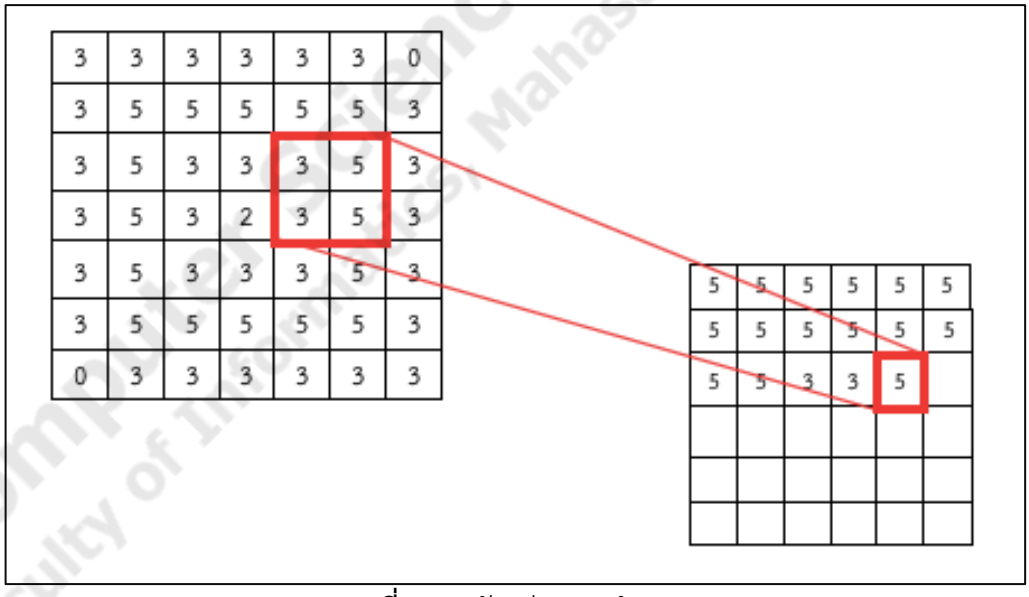
3	3					

ภาพประกอบที่ 3.17 ค่าใหม่หลังจากการ Stride ในรอบที่ 2

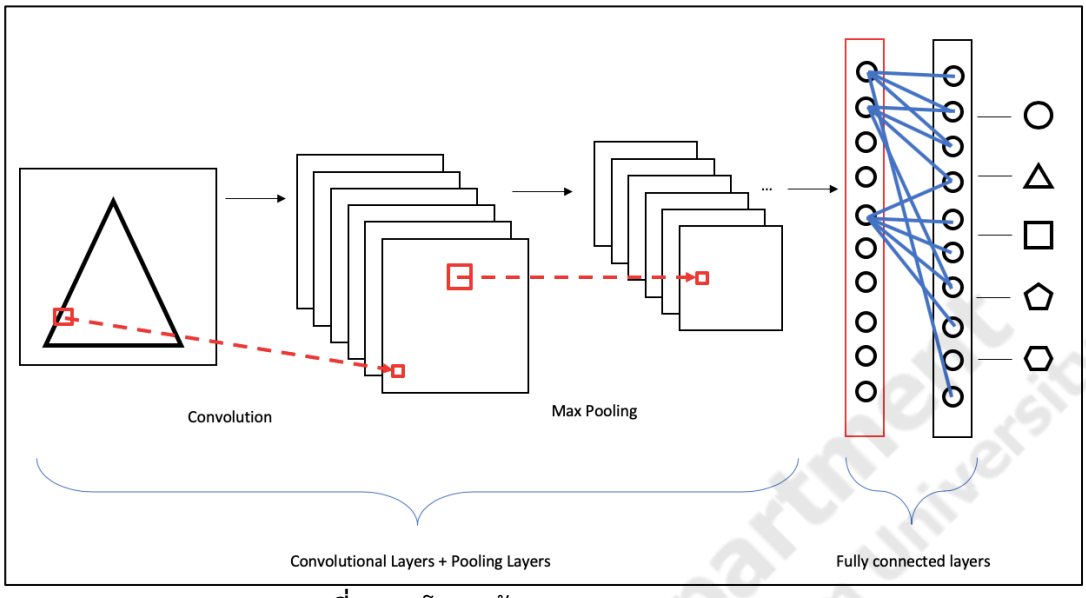
3	3	3	3	3	3	0
3	5	5	5	5	5	3
3	5	3	3	3	5	3
3	5	3	2	3	5	3
3	5	3	3	3	5	3
3	5	5	5	5	5	3
0	3	3	3	3	3	3

ภาพประกอบที่ 3.18 ผลลัพธ์จากการทำ Convolution จนครบทุกช่อง

Pooling Layer ขั้นตอนนี้ จะเป็นขั้นตอนของการ Resize ข้อมูลให้มีขนาดเล็กลงแต่รายละเอียด Input ยังคงเดิมส่งผลให้เพิ่มความรวดเร็วในการคำนวณซึ่งจะใช้ Max-Pooling ในการหาจากการทำข้างต้น กำหนด Stride เท่ากับ 1 Padding เท่ากับ 1 และ Filter ขนาด 2*2 Max-pooling เป็นการเอาค่าที่มากที่สุดในการวาง Filter Stride ในแต่ละครั้ง ดังภาพประกอบที่ 3.19



ภาพประกอบที่ 3.19 ตัวอย่างการทำ Max-Pooling

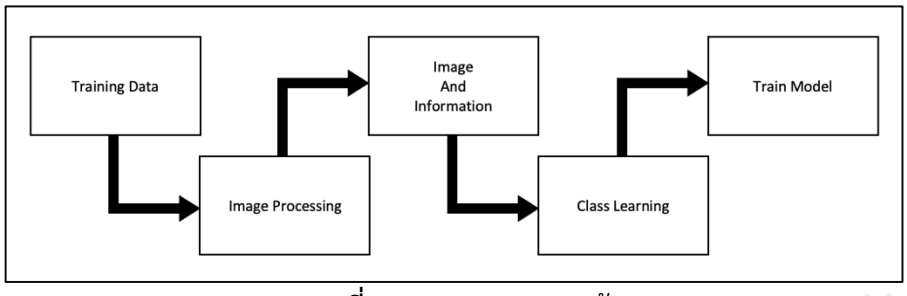


ภาพประกอบที่ 3.20 โครงสร้าง Convolution Neural Network

```
layers = [ ...
    imageInputLayer(imageSize,'Name','input')
    convolution2dLayer(5,10,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(5,20,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(5,40,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(5,80,'Padding','same')
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(40)
    reluLayer
    fullyConnectedLayer(20)
    reluLayer
    fullyConnectedLayer(10)
    fullyConnectedLayer(5)
    softmaxLayer
    classificationLayer ];
```

ภาพประกอบที่ 3.21 Code การกำหนด layers

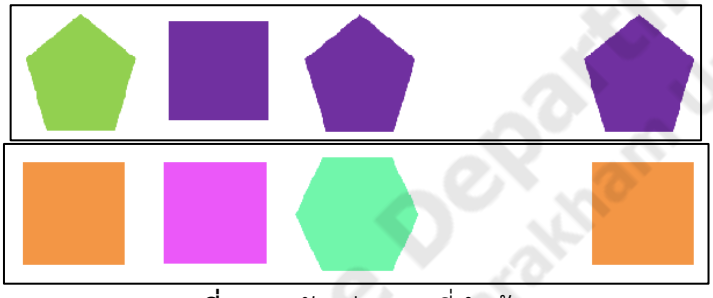
จากภาพประกอบที่ 3.20 ในส่วนของ Fully Connected layers จะมีส่วนของ Model ที่มีวิธีการสร้าง ดังภาพประกอบที่ 3.22



ภาพประกอบที่ 3.22 กระบวนการสร้างModel

Training Data

ชุดข้อมูลที่ใช้ในการ Training ดังภาพประกอบที่ 3.23

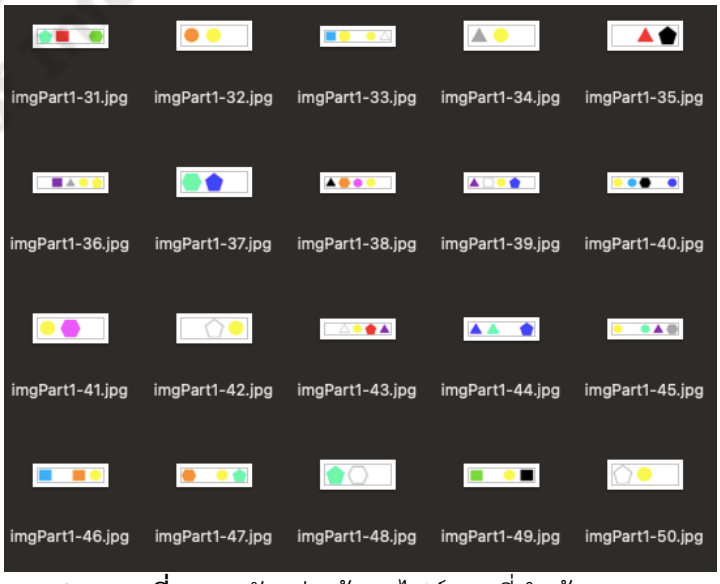


ภาพประกอบที่ 3.23 ตัวอย่างภาพที่นำเข้ามา Training

จากภาพประกอบที่ 3.23 จะมีภาพที่นำเข้ามา Training สามารถแบ่งเป็นกลุ่มได้ 5 กลุ่ม คือ วงกลม,สามเหลี่ยม,สี่เหลี่ยม,ห้าเหลี่ยม และหกเหลี่ยม

```
imds = imageDatastore('imagetrain',...
    'IncludeSubfolders',true,...
    'LabelSource','foldernames');
```

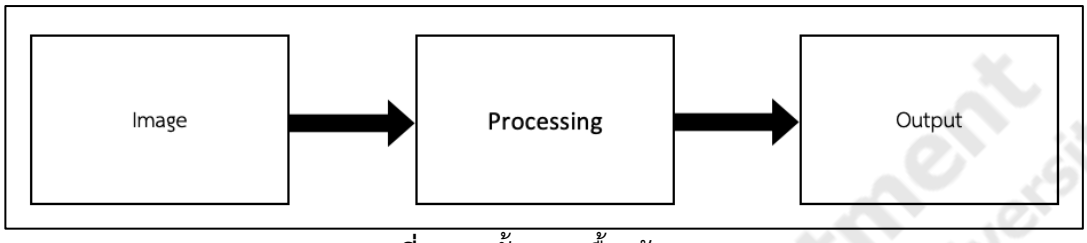
ภาพประกอบที่ 3.24 Code การโหลดข้อมูลภาพ



ภาพประกอบที่ 3.25 ตัวอย่างข้อมูลไฟล์ภาพที่นำเข้ามา Training

Image Processing

จากกระบวนการ Training Data จะได้ภาพที่สามารถเข้ากระบวนการ Image Processing ได้ โดยการนำภาพมาประมวลผล หรือคิดคำนวณด้วยคอมพิวเตอร์เพื่อให้ได้ข้อมูลที่เราต้องการทั้งในเชิงคุณภาพและปริมาณ การทำงานของ Image Processing จะมีขั้นตอนเบื้องต้น ดังภาพประกอบที่ 3.26



ภาพประกอบที่ 3.26 ขั้นตอนเบื้องต้น Image Processing

ในส่วนของ Processing ที่กล่าวข้างต้นมีอยู่มีกระบวนการหลักๆอยู่2กระบวนการ คือ

1. ReSize

   เป็นการกำหนดขนาดของภาพที่นำเข้ามา Training ให้มีขนาดที่เท่ากับ เพื่อให้ง่ายต่อการหาคำตอบ

2. เพิ่มขนาดเส้น

   เป็นการขยายเส้นของรูปทรงเรขาคณิต เพิ่มให้ง่ายต่อการจำแนก โดยการทำ Dilatation

การทำ Dilatation เพื่อทำการขยายส่วนของเส้นในภาพ ที่ทำต้องอยู่ในรูปแบบ Binary Image โดยใช้ตัวดำเนินการแบบ 4 ทิศทางของพิกเซลข้างเคียง

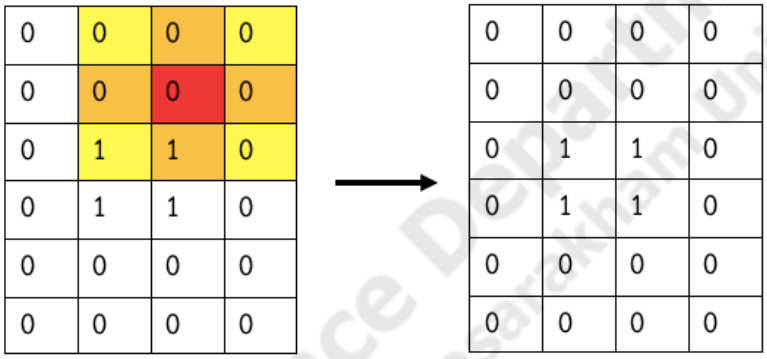
วิธีการ แสกนไปทุกข้อมูลบนภาพโดยใช้ตัวดำเนินการแบบ 4 ทิศทาง ของพิกเซลข้างเคียง ตลอดการแสกนถ้าจุดกลางของต้นแบบ มีค่าเท่ากับ 1 จะทำการยูเนียนระหว่างภาพต้นแบบและภาพที่ตำแหน่งและตรงกัน(เปลี่ยนค่าข้างเคียง 4 ทิศ โดยยึดค่าจุดกลาง) เพื่อทำการเปลี่ยนค่าจะส่งผลให้ภาพขยายใหญ่ขึ้น

0	1	0
1	1	1
0	1	0

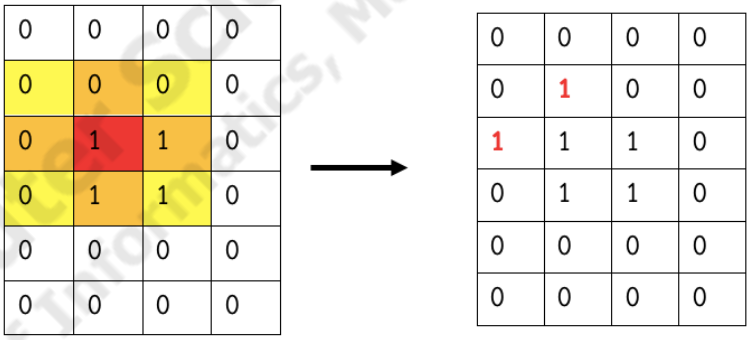
ภาพประกอบที่ 3.27 ตัวอย่างภาพ Template

0	0	0	0
0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0
0	0	0	0

ภาพประกอบที่ 3.28 ตัวอย่างภาพต้นฉบับก่อนทำ Dilatio



ภาพประกอบที่ 3.29 ตัวอย่างภาพก่อนและหลังทำ Dilation รอบที่ 2



ภาพประกอบที่ 3.30 ตัวอย่างภาพก่อนและหลังทำ Dilation รอบที่ 3

0	0	0	0
0	1	1	0
1	1	1	1
1	1	1	1
0	1	1	0
0	0	0	0

ภาพประกอบที่ 3.31 ตัวอย่างภาพหลังทำ Dilation

Image and Information

การเปรียบเทียบภาพจากขั้นตอน Image Processing ซ้ำไปเรื่อย ๆ จนกว่าภาพจะหมดจากจำนวนชุดข้อมูลของ Training Data และเก็บข้อมูลลงตัวแปลง

- Class Learning

เป็นการจำแนกประเภทข้อมูลของภาพที่เอามาจาก Training Data เพื่อแบ่งเป็น Class โดยแต่ละ Class จะแบ่งตามรูปทรงเรขาคณิต ดังภาพประกอบที่ 3.32

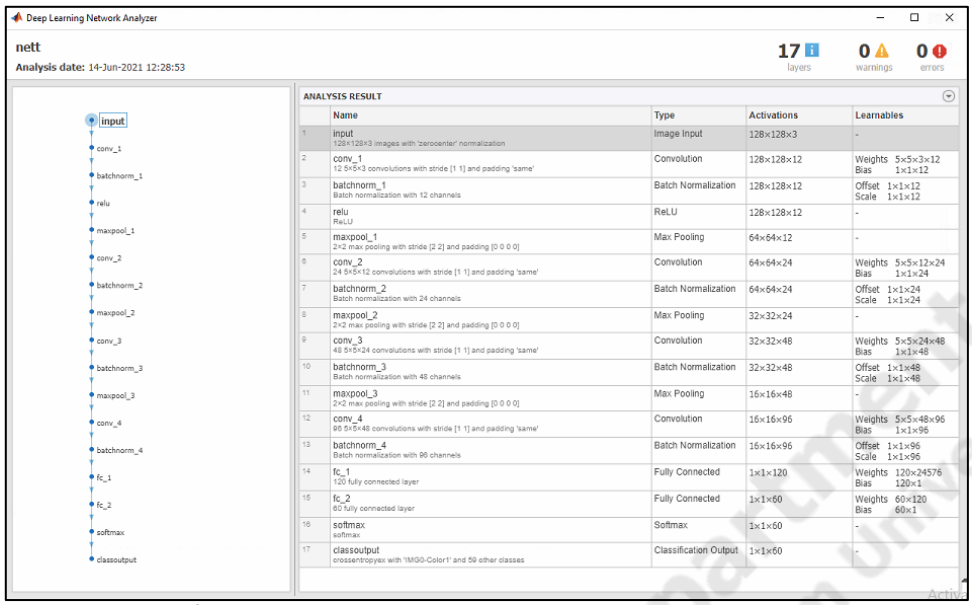
ภาพประกอบที่ 3.32 การแบ่ง Class ตามรูปทรงเรขาคณิต และ สี

Train Model

การสร้างโมเดลที่ได้ข้อมูลจาก Class Learning

1x1 ImageInputLayer
1x1 Convolution2DLayer
1x1 BatchNormalizationLayer
1x1 ReLULayer
1x1 MaxPooling2DLayer
1x1 Convolution2DLayer
1x1 BatchNormalizationLayer
1x1 MaxPooling2DLayer
1x1 Convolution2DLayer
1x1 BatchNormalizationLayer
1x1 MaxPooling2DLayer
1x1 Convolution2DLayer
1x1 BatchNormalizationLayer
1x1 FullyConnectedLayer
1x1 FullyConnectedLayer
1x1 FullyConnectedLayer
1x1 SoftmaxLayer
1x1 ClassificationOutputLayer

ภาพประกอบที่ 3.33 ตัวอย่าง Layers ของ model การรู้จำรูปเรขาคณิต



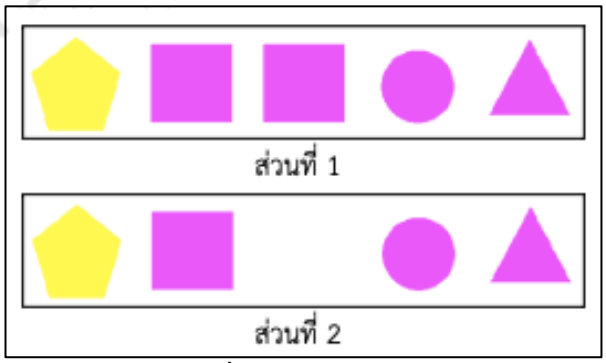
ภาพประกอบที่ 3.34 ตัวอย่าง การเรียง Layers ของ model การรู้จำรูปเรขาคณิต

3.5 การสุ่มรูปแบบคำถาม

การสุ่มรูปแบบคำถามจะได้มาจากการเรียงรูปทรงเรขาคณิต 5 รูป โดยใช้การสุ่มตัวเลขมาใช้แทนรูปทรงเรขาคณิต มีวิธีแทนดังนี้

- สุ่มได้หมายเลข 1 แสดงว่าได้รูปวงกลม
- สุ่มได้หมายเลข 2 แสดงว่าได้รูปสามเหลี่ยม
- สุ่มได้หมายเลข 3 แสดงว่าได้รูปสี่เหลี่ยม
- สุ่มได้หมายเลข 4 แสดงว่าได้รูปห้าเหลี่ยม
- สุ่มได้หมายเลข 5 แสดงว่าได้รูปหกเหลี่ยม

โดยเงื่อนไขการสุ่มนั้นจะสามารถ สุ่มซ้ำได้แค่ 3 ครั้ง หากเกิน 3 ครั้งให้ทำการสุ่มใหม่ และสุ่มตำแหน่งของตัวที่หายไป ดังภาพประกอบที่ 3.35



ภาพประกอบที่ 3.35 การสุ่มรูปแบบคำถาม

จากภาพประกอบที่ 3.35 จะมีการสุ่มตัวเลือกจากรูปแบบคำถาม ที่แบ่งเป็น 2 ส่วน โดยที่ส่วนที่ 2 จะมีรูปหายไปเกิดจากการสุ่ม

3.6 การหาคำตอบจากรูปแบบของคำถาม

การหาคำตอบจากรูปแบบของคำถาม เป็นการนำรูปส่วนที่ 2 เข้าไปทำการ testing ข้อมูลใน model ที่สร้างขึ้น จากนั้นหาคำตอบจาก classify ดังภาพประกอบที่ 3.36

```
ReIMG = imresize(newIMG,[64,64],'nearest');
[Pred,scores] = classify(net,ReIMG);
```

ภาพประกอบที่ 3.36 โค้ดของการหาคำตอบ

จากภาพประกอบที่ 3.36 จะต้องทำการ ReSize ขนาดของรูปให้เป็น [64 64] ก่อนจึงจะสามารถนำรูปนั้น มาหาคำตอบได้ โดยที่ Pred เป็นคำทำนายของคำตอบ และ scores เป็นเปอร์เซ็นต์ของการออกผลการทำนาย